CLAIMS

What is claimed is:

1. A method for preventing needless rescanning of objects comprising:

verifying an integrity of a file system by scanning the file system resulting in at least one known scanned region;

creating a database of the known scanned regions of the verified file system; and

validating an integrity of an object in the file system against the database of known scanned regions.

2. The method of claim 1, wherein verifying the integrity of the file system comprises scanning the objects in the file system for a presence of viruses.

3. The method of claim 1, wherein verifying the integrity of the file system further comprises monitoring the objects in the file system.

4. The method of claim 3, wherein monitoring comprises:

receiving a file system event from a real-time monitoring system, the file system event indicating that an object in the file system has been accessed; and

flagging the database of known scanned regions to indicate which of the known scanned regions was occupied by the accessed object.

5. The method of claim 4, wherein the database of known scanned regions comprises a copy of a partition table data structure indicating an identity and a location of a known scanned region occupied by the object.

6. The method of claim 5, wherein the partition table data structure includes an inode that contains information about the object other than name, and a directory block that contains the object name and a number of the inode of the object.

7. The method of claim 6, wherein the partition table data structure includes a super block that contains information about the file system as a whole, and a data block that contains a location in the file system where the object is stored.

8. The method of claim 7, wherein validating the integrity of an object comprises determining that the object does not occupy a flagged known scanned region.

9. The method of claim 6 wherein flagging comprises indicating which of the inodes and directory blocks were occupied by the accessed object.

10. The method of claim 9, wherein validating the integrity of an object comprises determining that the object does not occupy a flagged inode and directory block.

11. The method of claim 1 further comprising:

rescanning the object when the integrity of the object has not been validated; and

bypassing rescanning the object when the integrity of the object has been validated.

12. The method of claim 4, wherein the real-time monitoring system is a VxD program.

13. The method of claim 4, wherein the real-time monitoring system is a UNIX daemon.

14. The method of claim 4, wherein the real-time monitoring system is a network loadable module.

15. An apparatus comprising:

a machine-accessible medium having stored thereon executable instructions to cause a computer to perform:

verifying an integrity of a file system by scanning the file system resulting in at least one known scanned region;

creating a database of known scanned regions of the verified file system; and

validating the integrity of an object in the file system against the database of known scanned regions.

16. The apparatus of claim 15, wherein verifying the integrity of the file system comprises scanning every object in the file system for the presence of viruses.

17. The apparatus of claim 15, wherein verifying the integrity of the file system further comprises monitoring the objects in the file system.

18. The apparatus of claim 17, wherein the instructions for monitoring comprises:

receiving a file system event from a real-time monitoring system, the file system event indicating that an object in the file system has been accessed; and

flagging the database of known scanned regions to indicate which of the known scanned regions was occupied by the accessed object.

19. The apparatus of claim 18, wherein the database of known scanned regions comprises a copy of a partition table data structure indicating an identity and a location of the known scanned region occupied by the object.

20. The apparatus of claim 19, wherein the partition table data structure includes at least one of an inode that contains information about the object other than name, and a directory block that contains the object name and a number of the inode of the object.

21. The apparatus of claim 19, wherein the partition table data structure includes a super block that contains information about the file system as a whole, and a data block that contains a location in the file system where the object is stored.

22. The apparatus of claim 15, wherein validating the integrity of an object comprises determining that the object does not occupy a flagged region.

23. The apparatus of claim 18, wherein flagging comprises indicating which of the inodes and directory blocks were occupied by the opened or changed object.

24. The apparatus of claim 20, wherein the instructions for validating the integrity of an object comprises determining that the object does not occupy a flagged inode and directory block.

25. The apparatus of claim 15 further comprising instructions to cause a computer to perform:

rescanning the object when the integrity of the object has not been validated; and

bypassing rescanning the object when the integrity of the object has been validated.

26. The apparatus of claim 18, wherein the real-time monitoring system is a VxD program.

27. The apparatus of claim 18, wherein the real-time monitoring system is a UNIX daemon.

28. The apparatus of claim 18, wherein the real-time monitoring system is a network loadable module.

29. An integrity validator comprising:

a verifier to verify the integrity of a file system by scanning the file system resulting in at least one known scanned region;

a database of the known scanned regions of the verified file system; and

a validator to validate the integrity of an object in the file system against the database of known scanned regions.

30. The device of claim 29, wherein the verifier verifies the integrity of the file system by scanning the objects in the file system for the presence of viruses.

31. The device of claim 29, wherein the verifier monitors at least one of the objects in the file system.

32. The device of claim 31, wherein monitors comprises:

receiving a file system event from a real-time monitoring system, the file system event indicating that an object in the file system has been accessed; and

flagging the database of known scanned regions to indicate which of the known scanned regions was occupied by the accessed object.

33. The device of claim 32, wherein the database of known scanned regions comprises a copy of a partition table data structure indicating an identity and a location of a known scanned region occupied by the object.

34. The device of claim 33, wherein the partition table data structure includes an inode that contains information about the object other than name, and a directory block that contains the object name and a number of the inode of the object.

35. The device of claim 33, wherein the partition table data structure includes a super block that contains information about the file system as a whole, and a data block that contains a location in the file system where the object is stored.

36. The device of claim 29, wherein the validator validates the integrity of an object by determining that the object does not occupy a flagged known scanned region.

37. The device of claim 32, wherein flagging comprises indicating which of the partition table data structures were occupied by the accessed object.

38. The device of claim 34, wherein the validator validates the integrity of an object by determining that the object does not occupy a flagged partition table data structure.

39. The device of claim 29, further comprising:

an AV scanner to rescan the object when the integrity of the object has not been validated, wherein rescanning is bypassed when the integrity of the object has been validated.

40. The device of claim 32, wherein the real-time monitoring system is a VxD program.

41. The device of claim 32, wherein the real-time monitoring system is a UNIX daemon.

42. The device of claim 32, wherein the real-time monitoring system is a network loadable module.

43. A computer system comprising:

a processor coupled to a system bus;

a memory coupled to the processor through the system bus;

a machine-accessible medium coupled to the processor through the system bus;

an integrity process executed from the machine-accessible medium by the processor, wherein the integrity process causes the processor to verify the integrity of a file system resulting in at least one known scanned region, to create a database of known scanned regions of the verified file system, and to validate the integrity of an object in the file system against the database of known scanned regions.

44. The computer system of claim 43, wherein the system causes the processor to verify the integrity of the file system by scanning the objects in the file system for the presence of viruses.

45. The computer system of claim 44, wherein the system causes the processor to further verifies the integrity of the file system by monitoring the objects in the file system.

46. The computer system of claim 45, wherein the system causes the processor to monitor the objects in the file system by:

receiving a file system event from a real-time monitoring system, the file system event indicating that an object in the file system has been accessed; and

flagging the database of known scanned regions to indicate which of the known scanned regions was occupied by the accessed object.

47. The computer system of claim 46, wherein the database of known scanned regions comprises a copy of a partition table data structure indicating an identity and a location of a known scanned region occupied by the object.

48. The computer system of claim 47, wherein the partition table data structure includes an inode that contains information about the object other than name, and a directory block that contains the object name and a number of the inode of the object.

49. The computer system of claim 47, wherein the partition table data structure includes a super block that contains information about the file system as a whole, and a data block that contains a location in the file system where the object is stored.

50. The computer system of claim 43, wherein the system causes the processor to validate the integrity of an object by determining that the object does not occupy a flagged known scanned region.

51. The computer system of claim 46, wherein the system causes the processor to flag the database of known regions to indicate which of the inodes and directory blocks were occupied by the opened or changed object.

52. The computer system of claim 48, wherein the system causes the processor to validate the integrity of an object by determining that the object does not occupy a flagged inode and directory block.

53. The computer system of claim 43, wherein the system further causes the processor to:

rescan the object when the integrity of the object has not been validated; and

bypass the rescan of the object when the integrity of the object has been validated.

54. The computer system of claim 46, wherein the real-time monitoring system is a VxD program.

55. The computer system of claim 46, wherein the real-time monitoring system is a UNIX daemon.

56. The computer system of claim 46, wherein the real-time monitoring system is a network loadable module.